Flow Management in Packet Networks Through Interacting Queues and Law-of-Mass-Action Scheduling

Thomas Meyer and Christian Tschudin

Technical Report CS-2011-001
University of Basel
January 31st, 2011

Abstract

Classical queueing disciplines are work conserving: From the allocated channel's perspective this makes sense as the full bandwidth is exploited by sending packets back-to-back. But formally and traffic engineering wise, the resulting network dynamics is difficult to handle.

In this report, we show that deviating from this fundamental queuing assumption leads to much more controllable and analyzable forms of protocols. At the core of our work is a queue-scheduling discipline based on the chemical "Law of Mass Action" (LoMA) that serves a queue with a rate proportional to its fill level.

We introduce our LoMA-scheduling approach and provide a solid mathematical framework adopted from chemistry that simplifies the analysis of the corresponding queueing networks, including the prediction of the underlying protocols' dynamics. We demonstrate the elegance of our model by implementing and analyzing a TCP-compatible "chemical" congestion control algorithm C3A with only a few interacting queues (another novelty of our approach). We also show the application of our theory to gossip protocols, explain an effective implementation of the scheduler and discuss possibilities of how to integrate mass-action scheduling into traditional networking environments.

Keywords: fill-level-proportional rate scheduling, non-work-conserving forwarding, chemical networking protocols.

Universität Basel — Departement Mathematik und Informatik — Computer Networks Group
Bernoullistr. 16 | CH–4056 Basel | Switzerland Tel/Fax: +41 61 267 05 55/59 http://cn.cs.unibas.ch/

1 Introduction

A major problem in engineering communication protocols is to control the network-level dynamics caused by the protocols' implementations. While the early protocol designs in the 1960s and 70s focused on functional aspects, dynamics started to play a major role at MAC (Ethernet) and transport level (TCP) in the 1980s, queuing theory was adopted and congestion control was developed in parallel. However, till today it remains very hard to bridge the gap between the micro-behavior of arbitrary protocol logic and the macroscopical dynamics. For example, theoretical work like the *Network Calculus* by Le Boudec and Thiran (2004) helps to reason about the *worst-case* behavior of large work-conserving queueing networks, but is not able to characterize the *average* dynamic trajectory of a system. Ideally, an engineer would co-design functional and dynamic aspects of a new protocol. And conveniently, the protocol's dynamics would be expressed in terms of average flow properties and their robustness, rather that low-level queue parameters. In such an environment, the microscopic packet-delay details would depend on, and would be derived from, the macroscopic flow behavior, rather than being the starting point for an analysis.

To couple the micro- with the macro-level behavior is the core property of our "Law of Mass Action" (LoMA) approach. We will show that LoMA scheduling belongs to the class of non-work-conserving scheduling disciplines, which are rarely considered in computer networks. Non-work-conserving queuing adds delays to packets which, for efficiency reasons, seem to be a thing more to avoid rather than to embrace. Hence, before explaining LoMA in more details, we briefly review research that looked into non-work-conserving scheduling and in what context this was done.

1.1 Related Work

Several queueing disciplines proposed to introduce a small delay to packet streams. Delayed Frame Queueing (DFQ) (Le Pocher, Leung, & Gillies, 1999) for ATM networks sticks to work-conserving forwarding in the intermediate nodes but allows for non-work-conserving transmission at the network's edge. This leads to guaranteed upper bounds on delay and jitter for virtual circuits (VC), without relying on individual VC-queueing. Kamimura, Hoshino, and Shishikui (2008) also provide a controlled environment for jitter-sensitive IPTV streams. They proposed to equip Ethernet hubs with a Constant Delay Queueing (CDQ) policy. CDQ afflicts each packet of the constant-rate priority stream with a constant delay and forwards other best-effort packets during this waiting time. Wireless MAC protocols like 802.11 have many guard time intervals (because one cannot sense the wire as in 802.3), including random wait times, which leads to deliberate delays, too.

It has been shown that throughput-optimal allocation of the wireless channel capacity is achieved, in a work-conserving environment, with a maximum weight scheduling discipline that takes into account the queue fill-levels (e.g., in Maximum Weight Scheduling (Tassiulas & Ephremides, 1992), Maximum Weight Matching Scheduling (MWMS) or Queue Proportional Scheduling (QPS) (Eryilmaz, Srikant, & Perkins, 2001; Seong, Narasimhan, & Cioffi, 2006)). What is noteworthy is the prioritization of queues in proportion to their fill levels, which is similar to the behavior of our LoMA scheduling discipline.

A common element in these application fields is that they aim at making packet flows "more fluid", more predictable, and at providing strict or stochastic delay guarantees. However, to our knowledge, the corresponding non-work-conser-ving and fill-level-proportional scheduling disciplines are rarely discussed in the literature: At best they are mentioned as an exotic variant without being analyzed thoroughly (e.g., by van Mieghem, 2006, Chap. 13).

1.2 Law-of-Mass-Action scheduling in a nutshell

LoMA scheduling specifies in a rigid way the amount of delay to afflict to a packet such that some given macroscopic *rate* will result. More specifically, the LoMA states an inverse dependency between the fill level of a packet queue and the inter-packet service time: the more packets are in the queue, the quicker the system processes them. At the end, this means that the *waiting time* of a packet is constant, regardless of the number of packets in the queue. By designing local or remote packet processing loops and coupling them, complex dynamic behavior can be built, for example rate limiters, rate differentiators, integrators and much more. As we will show, this is already sufficient to design and formally analyze complex flow behaviors like gossip protocols or the well-known additive-increase / multiplicative-decrease congestion control as well as refinements thereof. Considering the fact that the end-to-end delay of packet flows is controllable and can be made constant when forwarded through LoMA-scheduled queues, we pave the path towards a scalable QoS infrastructure that operates without storing QoS state in the core, and in which flow aggregates have the same constant delay properties than individual flows.

1.3 Contributions

The *first contribution* of this paper is a technical as well as abstract framework based on interacting packet queues that permits to construct complex packet flows whose dynamic properties can be formally analyzed. This result rests on the import of concepts from chemistry, among them the mass-action scheduling regime, which is non-work-conserving (Section 2). Our *second contribution* is to show that LoMA scheduling can be done effectively, which means that one can "run" corresponding protocols in an effective way, turning our conceptual framework into an executable model that can operate at wire speed (Section 4). The *third contribution* lies in reformulating two well-known protocols (one from the world of gossip protocols, the other being the famous TCP protocol for which we provide a "chemical" congestion control algorithm that is TCP fair) and to analyze their well-known dynamics in new and elegant forms (Section 5). *Forth*, we discuss how (classical) packet flows can be steered by our chemically-inspired approach using a chemically modeled control plane and how compatible our packet flows are with the existing packet-forwarding infrastructure (Section 6).

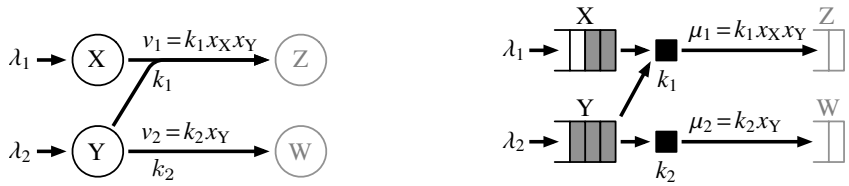
2 Mass-Action Scheduling

In this section, we describe the law of mass action from chemistry and express it in terms of the well-known M/M/1 queueing model. We start by showing how to map concepts from chemistry ("molecular reactions") to events on packet queues.

2.1 Molecules as Packets, Reactions as Queues

Figure 1 shows on the left side a system of two chemical reactions r_1 and r_2 having two reactants X and Y and two products Z and W. More precisely, r_1 is a bimolecular reaction $X + Y \xrightarrow{k_1} Z$ while the competing reaction r_2 simply turns a Y-molecule into a W-molecule. The rate at which these reactions happen is controlled by the respective coefficients k_1 and k_2 called rate constants.

We map above configuration to a system of two queues X and Y (Figure 1, right side) and make an analogy between chemistry and networking by treating packets as molecules. A molecular species can be viewed as a buffer that temporarily stores molecule instances until they are being consumed by an egress reaction. In terms of networking, we have two packet queues X and Y served by two servers. The upper



(a) Abstract chemical reaction model

(b) Corresponding network of FIFO queues

Figure 1: Bimolecular reaction in two different settings – A bimolecular reaction extracts an instance of two chemical species (= packet queues), X and Y. At the same time, the queue Y is drained by a second server (= monomolecular reaction).

server is special in that it extracts a packet from two queues at the same time. Table 1 summarizes how we translate chemical ideas to queueing theory.

Chemical metaphor…	**…for**	**Symbol**
Molecule instance	Packet	
Molecular species	Queue	$X, Y, Z, \ldots$
Concentration	Fill level	$x_X, x_Y, x_Z, \ldots$
Reaction	Server	$r_1, r_2, r_3, \ldots$
Reaction rate	Service rate	$v_1, v_2, v_3, \ldots$
Inflow	Fill rate	$\lambda_1, \lambda_2, \lambda_3, \ldots$

Table 1: Chemical queueing metaphor

2.2 The Law of Mass Action (LoMA)

The "Law of Mass Action" stipulates that the reaction rate depends proportionally to the abundance of the involved reactants. This principle is rooted in chemical kinetics, which explains the random movement of molecules by Brownian motion. The more molecules a fixed volume contains, the more frequently they collide and react in turn. At the macroscopic level this manifests in the law of mass action (Waage & Guldberg, 1864; english translation by Abrash, 1986).

In Figure 1, we write that the production rate of r_1 is (according to the LoMA) $v_1 = k_1 x_X x_Y$, where x_X and x_Y are the concentrations of the molecules X and Y. The rate $v_2 = k_2 x_Y$ of the decay reaction r_2 also depends on the concentration, namely on that of Y alone, and is linear.

Figure 1 is completed by also showing the arrival rates λ_1 and λ_2 which, on the networking side, express how fast packets are put into the corresponding queues, while the concentrations x_X and x_Y, again on the networking side, become the fill level of the (potentially infinite) queues X and Y.

Before we introduce our artificial packet chemistry formally in the next section, in the following, we first highlight and discuss the main properties of LoMA-scheduled queues.

2.3 Properties of LoMA-Scheduled Queues

Like the classical M/M/1 queue, our LoMA-scheduled queue is modeled as a Markov birth-death chain. As depicted in Figure 2 and unlike for the classical queue, the mean death transition rate is proportional to the fill level. It is like waiting packets build up a "pressure" that accelerates the service rate of the queue.

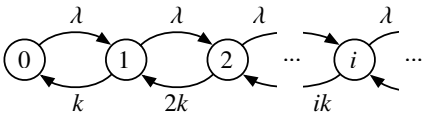


Figure 2: Markov birth-death chain of a LoMA-scheduled queue.

As listed in Table 2 and shown in detail later (Sections 3.3 and 4.2), the expected number of packets in the queue, $\hat{x}$, is proportional to the packet arrival rate λ. This leads to a constant waiting time T in steady state, meaning that the *latency* of a packet is independent on the arrival rate and independent on the fill level. These two quantities are related by Little's Law, which still applies for the mass-action scheduling regime.

Property of Queue	**M/M/1**	**LoMA**
Expected fill level $\hat{x}$	$\lambda/(\mu-\lambda)$	λ/k
Expected waiting time T	$1/(\mu-\lambda)$	$1/k$

Table 2: Queue properties – Properties of the traditional M/M/1 queue compared to a LoMA-scheduled queue.

2.4 Discussion

FIFO queues are ubiquitous in packet networks and mandatory for statistical multiplexing. Placed in front of limited resources, such as bandwidth-limited links, they buffer bursts of packets that would otherwise be lost. Most often the M/M/1 queueing model is used, in which the arriving traffic is assumed to be a Poisson process with mean arrival rate λ, and where there is one server that drains the queue at constant mean rate μ. A constant service rate is only reasonable if two assumptions hold: First, the server must be triggered by a finite resource, such as a link, for which the queue acts as a buffer. The second assumption is that of *work conservation*: If there is a packet in the queue and if the resource is idle, the server will immediately drain a packet from the queue. In this paper, we relax the second assumption and look for *non-work-conserving scheduling rules* where the server may stay idle although the queue contains packets. This will be discussed in Section 4.1.

An intentional departure from the real (chemical) world is that in our model we will keep track of molecule instances in order to maintain the packet order; that is, we stick to the FIFO order in packet queues.

A bimolecular reaction aggregates the consumed packets. Similar to a transition in a Petri Net, such a server synchronizes multiple packet streams, as a packet in one queue can only be processed when there are packets in the other reactant queues. Thus synchronization allows one packet flow to control another flow of packets.

Finally we note that in our model a queue may be drained by multiple servers: Reaction r_2 in Figure 1 for example drains packets from queue Y independent of reaction r_1. This requires that the competing servers coordinate their action. As we will see in Section 4.1, the servers are communicating indirectly through the fill level of the queue.

3 Artificial Packet Chemistry

The power of our proposed networking model relies partly on the mass-action scheduled queues but also rests on the combination of queues – that is, on the concept of chemical reactions. By coupling

mass-action scheduled queues we build distributed queueing networks that have different properties than networks of classical queues. For example, they promote stable equilibrium solutions and are easier to analyze. Furthermore, with the help of multi-molecular reactions, packet streams are able to control the rate of other packet streams, enabling a new class of distributed algorithms that exploit dynamic packet-stream interactions in order to compute a distributed result, just as ordinary communication protocols also do.

We start this section by introducing *Disperser* – a simple distributed "chemical" algorithm, which also has a classical protocol formulation. In Section 3.2, we then introduce a formal model for describing distributed LoMA-scheduled queueing networks before we discuss formal methods to analyze such networks in Section 3.3. Section 3.4 applies one of these methods to our *Disperser* example in order to formally prove its convergence.

3.1 Consensus and Distributed Computation Through Interacting Packet Flows

As a simple introductory example we present *Disperser* (Meyer & Tschudin, 2009) – a simple LoMA-scheduled queueing network that computes the average of distributed values. Its operation principle is directly comparable to gossip-based or epidemic protocols like the *Push-Sum* protocol (Kempe, Dobra, & Gehrke, 2003), which perform distributed computation by disseminating information to randomly selected neighbors.

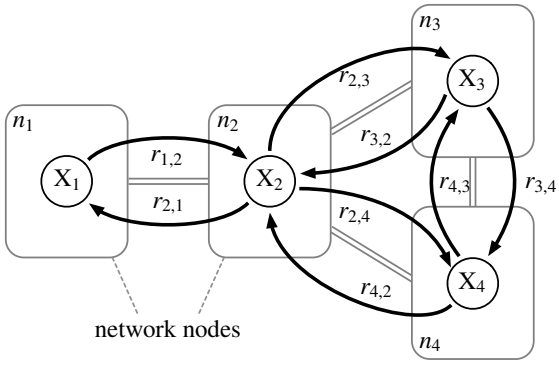


Figure 3: Example Network for the *Disperser* Protocol – A distributed LoMA-scheduled queueing network that globally averages the fill levels of the queues.

Figure 3 depicts *Disperser's* reaction network for a four-node topology that consists of one packet queue per node. Each queue is drained in parallel by one separate server per network link. In chemical terminology, there is one reaction per node-link pair that consumes a local molecule and produces it in the corresponding neighbor node.

If we would run this queuing network with traditional work-conserving scheduling, molecules would accumulate at the node with the highest degree, as all other nodes are faster to shuffle packets to it. However, with our LoMA scheduling, when queues are scheduled according to the law of mass action, the system has a global attractor that is independent of the topology. Figure 4 shows a typical simulation run, as well as a one-to-one comparison with a corresponding *Push-Sum* implementation, which has the same dynamics. At the equilibrium state, each queue contains the average number of packets in the system. Thus our simple distributed LoMA-scheduled queueing network disperses the packets uniformly across the network, hence its name.

Interestingly and typically for "chemical" algorithms, the computation is not carried out symbolically

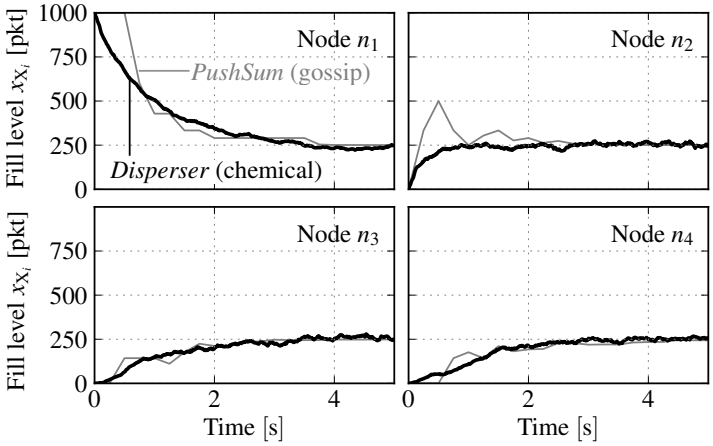


Figure 4: OMNeT++ simulation of *Dipserser* and *Push-Sum* – The queueing network has a global attractor where the packets are distributed uniformly across the queues (here for the topology depicted in Figure 3).

by explicitly calculating "the average". The result is rather emerging from the dynamic interaction of packet flows. Protocol states are represented by the multiplicity of packets in queues whereas information is conveyed to remote nodes via the packet *rate*. The law of mass action plays an important role by mapping fill levels to packet rates and vice-versa.

In the following, we will introduce a formal model of LoMA-scheduled queueing networks and their analysis in general before we provide a simple convergence proof of *Disperser* in Section 3.4.

3.2 An Abstract Formal Model of LoMA-Scheduled Queueing Networks

We introduce an *artificial packet chemistry* as a formal model to globally define the queues and their interconnections in a given computer network. An artificial packet chemistry is defined[1] as quadruple $\mathcal{PC} = (\mathcal{G}, \mathcal{S}, \mathcal{R}, \mathcal{A})$. The network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents the computer network, where $\mathcal{V} = \{n_1, \ldots, n_{|\mathcal{V}|}\}$ is the set of all nodes representing "chemical reaction vessels", and where $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ are unidirectional network links connecting neighbor nodes.

The set of molecular species $\mathcal{S} = \cup_{i \in \mathcal{V}} \{\mathcal{S}_i\}$ lists the queue instances $\mathcal{S}_i$ of all nodes $i \in \mathcal{V}$, whereas the set of reaction rules $\mathcal{R} = \cup_{i \in \mathcal{V}} \{\mathcal{R}_i\}$ lists all flow relations among the local queues in node i. Each reaction rule $r \in \mathcal{R}_i$ is characterized by a reactant and a product multiset and is given by the chemical reaction equation

$$r \in \mathcal{R}_i: \quad \overbrace{\sum_{s \in \mathcal{S}_i} \alpha_{s,r} s}^{\text{reactants}} \longrightarrow \overbrace{\sum_{\{s \in \mathcal{S}_i \cup \{\mathcal{S}_j\} \mid j \in \mathcal{N}_i\}} \beta_{s,r} s}^{\text{products}} \tag{1}$$

where $\mathcal{N}_i$ is the set of neighbors of node i. The stoichiometric coefficients $\alpha_{s,r} \in \{0, 1\}$ and $\beta_{s,r} \in \{0, 1\}$ denote whether a packet in queue s is consumed and/or produced by reaction r, respectively[2]. While all reactant species must reside in the same node in order to react, the product species may either be local species or species in the reactant's direct neighborhood. This is how the chemical model represents packet transmission – as reactions across neighboring vessels.

[1] This definition extends the definition of a generic *artificial chemistry* (Dittrich, Ziegler, & Banzhaf, 2001) by a network topology.

[2] In chemistry, reactions may produce/consume more than one instance of each species ($\alpha_{s,r}, \beta_{s,r} \in \mathbb{N}_0$).

Finally, the reaction algorithm $\mathcal{A}$ defines when which reaction occurs. According to the law of mass action, the *mean reaction rate* v_r of each reaction $r \in \mathcal{R}$ depends on the left-hand side of the reaction equation and is proportional to the fill level x_s of the reactant queues $s \in \mathcal{S}$ if $\alpha_{s,r} = 1$:

$$v_r = k_r \prod_{s \in \mathcal{S}} x_s^{\alpha_{s,r}} \tag{2}$$

There are many efficient implementations of mass-action schedulers, such as Gillespie's Direct Method (1977) or Gibson and Bruck's Next Reaction Algorithm (2000). So far they served biologists to simulate real chemical reactions, but they can directly be applied to trigger the servers of networking queues. We will sketch a typical implementation later in Section 4.1.

Example. According to our definition, the *Disperser* example in Figure 3 is defined as artificial packet chemistry $\mathcal{PC} = (\mathcal{G}, \mathcal{S}, \mathcal{R}, \mathcal{A})$ where the network graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains four nodes $\mathcal{V} = \{n_1, \ldots, n_4\}$ and is connected by the edges $\mathcal{E} = \{(n_1, n_2), (n_2, n_1), (n_2, n_3), (n_3, n_2), (n_2, n_4), (n_4, n_2), (n_3, n_4), (n_4, n_3)\}$. The set of species contains one queue per node: $\mathcal{S} = \{\mathrm{X}_1, \ldots, \mathrm{X}_4\}$, and the set of reactions consists of one reaction per unidirectional link: $\mathcal{R} = \{r_{a,b} \colon \mathrm{X}_a \longrightarrow \mathrm{X}_b \mid (a, b) \in \mathcal{E}\}$.

3.3 Formal Analysis of LoMA-Scheduled Queueing Networks

One advantage of our chemical queuing model is that we can apply a vast number of analysis methods for chemical reaction kinetics directly to queuing networks on different scales and granularity. We show this from microscopic to macroscopic layers along six topics: We briefly discuss (1) stochastic and (2) deterministic mathematical models of chemical queueing network dynamics. (3) We then show how this translates to intuitive rules to design and study such networks, before we (4) show how to analyze their transient behavior. Furthermore, (5) stochastic and deterministic descriptions can be combined, and (6) there are generic high-level methods to determine the stability of reaction networks based only on their topology.

Microscopic stochastic analysis. The dynamic behavior of chemical reaction networks is described by the Chemical Master Equation (CME) (McQuarrie, 1967), which can be derived directly from collision theory (Gillespie, 1992). The master equation describes a continuous-time discrete-space Markov jump process on the $\mathcal{S}$-dimensional integer lattice $\mathbb{N}_0^{|\mathcal{S}|}$, which is similar to that of classical queueing networks, but with fill-level proportional transition rates (see Figure 2).

The following general statements about the statistical distribution of molecules in a chemical reaction network are known: For closed first-order reaction networks (i.e., when the total number of packets is bound and the network only contains unimolecular reactions) the steady-state probability fill-level distribution is multinomial (Gadgil, Lee, & Othmer, 2005; Jahnke & Huisinga, 2007). For open first-order reaction networks, the steady-state probability distribution is in product form following a Poisson distribution (Gadgil et al., 2005; Jahnke & Huisinga, 2007); this also holds if the reaction network contains bimolecular reactions (Gelenbe, 2008).

Macroscopic deterministic analysis. Like for classical queuing theory, a detailed stochastic treatment is only feasible for simple networks. Chemists also rarely study the dynamics of reaction networks at the microscopic stochastic level. Instead, they approximate the average trajectory of the system by Ordinary Differential Equations (ODE) – one equation for each species. This also has a tradition in networking,

where a fluid model is often built in order to analyze the dynamics of protocols and packet flows (Misra, Gong, & Towsley, 2000; Shakkottai & Srikant, 2002; Low, 2003). The problem of fluid network models is that they have to be extracted manually from the protocol's source code and that it is not always clear whether they accurately model the real behavior.

For chemical queueing networks the fluid model is accurate and easy to obtain: Gillespie linked the stochastic master equation of generic chemical reaction networks to different simplified models including the Chemical Langevin Equation (Gillespie, 2000), a Fokker-Planck approximation (Gillespie, 2002), and a deterministic description based on ODEs. He showed that the ODE approximation accurately predicts the systemic trajectory of the system for most of the cases (Gillespie, 2000). Since the queueing network actually performs a distributed simulation of a chemical reaction system, the ODE-model can be generated directly and automatically from the topology of the corresponding reaction network.

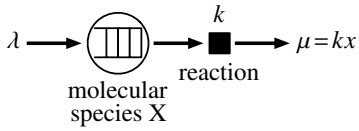


Figure 5: A chemical species acts as a packet queue.

As an example of a fluid model of a single queue, consider Figure 5. The fill level x of queue X is increased at rate λ and decreased at rate μ, which yields the differential equation

$$\dot{x} = \overbrace{\lambda}^{\text{inflow}} - \overbrace{kx}^{\text{outflow}} \tag{3}$$

Since the outflow rate is proportional to the fill level x there is always a stable fixed point: For $\dot{x} \equiv 0$, we obtain a steady-state fill level of $\hat{x} = \lambda/k$ (also compare to Table 2).

Kirchhoff's current law. The LoMA scheduling leads to further observations on the macroscopic flow level that simplify the intuitive understanding of a queuing network by just looking at the corresponding reaction network graph. One such observation is Kirchhoff's current law in electronic circuits, which also holds for chemical reaction networks at equilibrium (Perelson & Oster, 1974). It states that the total rate of packets flowing into a queue is equal to the total rate of packets drained from the queue. This law can be derived naturally from the queue's ODE and is very helpful in designing and analyzing chemical queueing networks with pen and paper.

Transient analysis. There are well-known methods to analyze the transient behavior of chemical reaction networks that can be applied directly to LoMA-scheduled queueing networks: The Metabolic Control Analysis (Reder, 1988; Heinrich & Schuster, 1996; Fell, 1997; Hofmeyr, 2001) studies the sensitivity of states (fill levels) to internal and external perturbations. The transient behavior of fill levels and packet flows can also be analyzed by signal and control theory (Ingalls, 2004). Perturbation analysis, for example, linearizes the ODE system around the fixed point and studies the system's stability or frequency response.

Mesoscopic analysis. In order to estimate the variance of fill levels (i.e., the intrinsic noise caused by the stochastic process) researchers proposed mesoscopic analysis methods, where the mathematical description of the stochastic process is separated into a systemic part and the augmented noise. The

queueing network can for example be described by Chemical Langevin Equations (Gillespie, 2000), or modeled by the Linear Noise Approximation (van Kampen, 1976; Elf & Ehrenberg, 2003; Tomioka, Kimura, Kobayashi, & Aihara, 2004; van Kampen, 2007) or the Two Moment Approximation (Gómez-Uribe & Verghese, 2007; Ullah & Wolkenhauer, 2009a, 2009b).

Generic stability rules. In the past decades, research on chemical reaction networks focused on macroscopic properties of complex reaction networks. There are for example different methods to predict the stability of reaction networks by just studying their topology – that is, without the need to analyze the ODEs: The Deficiency Zero Theorem (Feinberg, 1972; Horn & Jackson, 1972; Horn, 1973a, 1973b; Mairesse & Nguyen, 2009; Anderson, Craciun, & Kurtz, 2010) provides conditions for reaction networks to exhibit a single stable fixed point. Dittrich's Chemical Organization Theory (COT) (Dittrich & Speroni di Fenizio, 2007), on the other hand, identifies *organizations* as sets of queues that forward traffic in steady state. Organizations are related to quasi steady states of the underlying Markov process, but are easier to obtain. A queueing network may have different organizations – that is, different stable flow patterns through different sets of queues. The COT then proves that only such self-organizing queuing networks are dynamically stable.

3.4 Formal Convergence Proof with a Perturbation Analysis

The ODE description allows for analyzing a mass-action-scheduled queueing network with a classical perturbation analysis. We now prove that our example *Disperser* algorithm (see Figure 3) always converges to the stable fixed point where each queue contains the average number of packets in the network. We show this for arbitrary network sizes *and* arbitrary topologies if the network is (a) symmetric and (b) connected. In general, such a convergence proof is carried out in three steps: First, we write down the differential equations of all queues; second, we find the global fixed point and show that the packets are uniformly distributed; and third, we prove that this fixed point is asymptotically stable.

1. Each queue receives packets from its neighbors at a rate equal to the fill level the neighbor's queue. Concurrently, one reaction per neighbor drains the local queue. The differential equation describing the fill level x_i of queue X_i in node i is

$$\dot{x}_i = \overbrace{\sum_{k \in \mathcal{N}_i} x_k}^{\text{inflow}} - \overbrace{\deg(i) x_i}^{\text{outflow}} \quad \forall i \in \mathcal{V} \tag{4}$$

   where $\deg(i)$ is the degree[3] of node i, and where the set $\mathcal{N}_i$ contains the neighbors of node i.

2. We find the fixed point by setting all $\dot{x}_i \equiv 0$:

$$\hat{x}_i = \frac{\sum_{k \in \mathcal{N}_i} \hat{x}_k}{\deg(i)} \quad \forall i \in \mathcal{V} \tag{5}$$

   That is, the equilibrium fill level of queue X_i is equal to the average fill level in the node's neighborhood. In a connected network this only holds iff all queues contain the same number of packets: $\hat{x} = \hat{x}_i$. Since all reactions conserve the total number of packets ($x_{\text{tot}} = \sum_{i \in \mathcal{V}} x_i$) the fill level of all queues is equal to the average number of packets in the system.

[3] Since we require the network to be symmetric, the in-degree of a node is equal to its out-degree.

$$\hat{x} = \hat{x}_i = \frac{x_{\text{tot}}}{\left|\mathcal{V}\right|} \quad \forall i \in \mathcal{V} \tag{6}$$

3. This fixed point is asymptotically stable if the system returns back after a small perturbation. For the corresponding analysis we calculate the $\left|\mathcal{V}\right| \times \left|\mathcal{V}\right|$ Jacobian matrix of (4):

$$\mathbf{J} = \left[j_{i,k}\right] = \left[\frac{\partial \dot{x}_i}{\partial x_k}\right] = -\mathbf{L}(\mathcal{G}) \tag{7}$$

where $\mathbf{L}(\mathcal{G})$ is the Laplacian of the network graph

$$\mathbf{L}(\mathcal{G}) = \left[l_{i,k}\right] = \begin{cases} \deg(i) & \text{if } i = k; \\ -\operatorname{adj}(i,k) & \text{otherwise.} \end{cases} \tag{8}$$

The adjacency function $\operatorname{adj}(i,k)$ yields the value 1 if node i is connected to node k, or 0 otherwise. Since any Laplacian of a connected symmetric graph has positive real eigenvalues, the eigenvalues of (7) are negative and the fixed point in (6) is stable for arbitrary network topologies. □

Compared to the convergence proof of *Push-Sum*, our proof is considerably more compact. Convergence proofs are often simpler in a LoMA-scheduled queueing network because we can automatically generate a fluid model from the structure of the protocol's queueing network and apply a classical perturbation analysis.

4 Execution Models and Reaction Network Design

The artificial packet chemistry introduced in the previous section serves as an abstract model to design and analyze LoMA-scheduled queuing networks. In order to build real hard- and/or software that closely follows this abstract model, we have to address engineering concerns and implementation issues, such as how the proclaimed macroscopical LoMA rate can be achieved by a microscopic packet-per-packet scheduling algorithm.

In Section 4.1, we first discuss the implementation of an efficient mass-action scheduler, before we present an execution engine that is able to "run" packet reactions. Because the engine's dynamic behavior is predicted by the abstract model, we are able to close the gap between a protocol's microscopic implementation and its macroscopic behavior. Section 4.2 proposes a separation between traffic forwarding engine and "chemical" control layer that can automatically be compiled from the abstract model. The chemical networks presented so far have been static: In Section 4.3 we show another execution method based on active packets (Fraglets) which permits to reshape the queueing network at run-time with an active networking approach. Finally, Section 4.4 presents design patterns that are useful to synthesize "chemical" protocols.

4.1 An Efficient Mass-Action Scheduler

Each network node runs a scheduler that performs the following tasks: The scheduler computes the next occurrence time of each server $r \in \mathcal{R}_i$ in the local node i according to the law of mass action, sorts these events into a priority queue, waits until the first event occurs, and executes it. The difficulty is to dynamically re-schedule the events if packets are added or drained from the dependent queues. As

demonstrated by Gibson and Bruck (2000), it is possible to implement an efficient mass-action scheduler that only requires $O(\log|\mathcal{R}|)$ time to enqueue or dequeue packets. In the following, we give an outline of Gibson and Bruck's Next Reaction Method (2000) and a possible implementation for queueing networks.

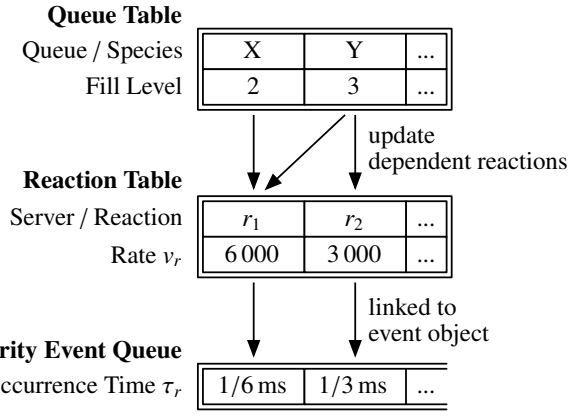


Figure 6: Reaction Scheduler Implementation – The content of the data structures corresponds to the example reaction network and state depicted in Figure 1 for $k_1 = 10^3/(\text{pkt s})$ and $k_2 = 10^3/\text{s}$.

Data structures. Figure 6 depicts the three data containers our implementation maintains. Each queue $s \in \mathcal{S}$ keeps track of its fill level and informs the dependent server. Each server $r \in \mathcal{R}$ then computes and stores the target rate v_r according to the law of mass action (2). Generally, the interval at which the servers drains the queues is inversely proportional to this rate: $\tau_r = 1/v_r$. Each server initially registers an event for time τ_r with the local event scheduler, which is implemented as an indexed priority queue. This queue sorts the event, earliest time first. If the queue is implemented as a tree, this can be done in $O(\log|\mathcal{R}|)$ time (Gibson & Bruck, 2000).

Dynamic scheduling. The scheduler has to reshuffle the events dynamically as packets enter or leave the queues. For example, if a server is forced to wait 1 ms but a packet arrives within this period, the service interval has to be shortened.

Let us illustrate the scheduling process based on the simple queueing network depicted in Figure 1: Figure 6 shows the state of the scheduler at time $t_{\text{now}} = 0\,\text{s}$, assuming that there are two packets in queue X and three packets in queue Y, and that the reaction constants are $k_1 = 10^3/(\text{pkt s})$ and $k_2 = 10^3/\text{s}$, respectively. Consequently, the first server will serve its queues (before the second) at time $t = \frac{1}{6}$ ms, as shown at the top of Figure 7. The scheduler picks the first event from the priority queue and waits until its occurrence time is reached. It then triggers the corresponding server, which extracts packets from the reactant queue(s) and injects them to the product queue(s). Thereafter, the server recomputes its interval and reschedules its event.

In our case, reaction r_1 removes a packet from both queues X and Y. Since the occurrence time of reaction r_2 also depends on the fill level of queue Y, it is necessary to reschedule this reaction, too. Two packets are left in queue Y, so the new reaction rate of r_2 is $v_{r_2,new} = 2 \times 10^3\,(\text{pkt/s})$. Because the occurrence time of r_2 has not elapsed yet, the scheduler has to rescale it according to the following equation:

$$\tau_{r,new} = \overbrace{\frac{v_{r,old}}{v_{r,new}}}^{\text{scaling}} \cdot \overbrace{\left(\tau_{r,old} - t_{\text{now}}\right)}^{\text{time remaining}} + t_{\text{now}} \tag{9}$$

The bottom part of Figure 7 shows how the occurrence time of r_2 is prolonged a bit in turn.

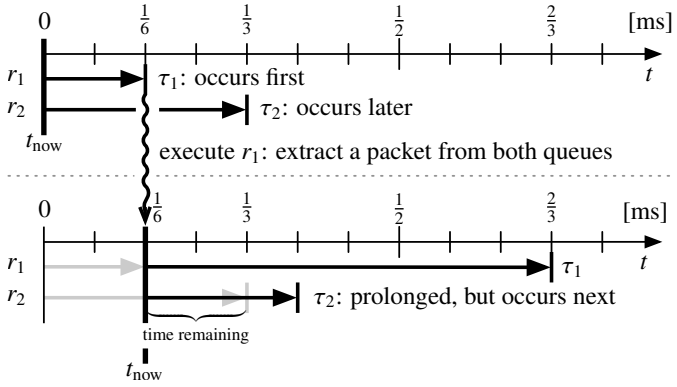


Figure 7: Dynamic Scheduling – Situation before and after the occurrence of the first reaction. Time intervals of reactions are rescaled when the fill level of dependent queues changes.

The same figure also illustrates that our scheduling algorithm is not a priority scheduler: the reactions are interleaved according to their weight (=rate), meaning that the reaction with the highest rate is not always scheduled first. At the macroscopic flows level, this scheme leads to the correct expected packet flow rates according to the law of mass action.

Note that unlike for Earliest Deadline First (EDF) or similar timed scheduling algorithms, our approach does not require tagging each packet with a timestamp. This is because the service rate is proportional to the fill level of all dependent queues, which means that the superposition principle applies: we can treat all packets in a queue together and only schedule one event per reaction.

4.2 A Chemical Flow Control Plane

LoMA-scheduled queueing networks may be used to perform traffic shaping. In such a scenario, the queueing network can be modeled as black-box system with multiple input and output packet flows. The system applies a dynamic filter (in the form of a chemical reaction network) to the "rate signal" of the traversing packet flows and reshapes their traffic patterns.

Instead of sending data packets through a complex queueing network in order to shape the packet flow, we envision an execution model that separates packet forwarding from "chemical" flow control as depicted in Figure 8: An ingress packet flow i ($1 \leq i \leq n$) is sent to a separate FIFO queue that is drained by one server each. The servers have no predefined service rates; instead the rates are determined dynamically by the chemical control plane, where each queue X_i is represented by an input species X_i and its server by an output species X_i^*. Between these species the traffic engineer can span an arbitrary chemical reaction network, which may also be influenced by the activity of the other flows.

Whenever a data packet enters a queue, a molecule is injected to the corresponding input species in the control plane. The reaction network in the chemical control plane simulates the network with a mass-action scheduler and eventually generates an instance of the output species. This immediately triggers the corresponding server in the forwarding plane and causes the next packet to be dequeued.

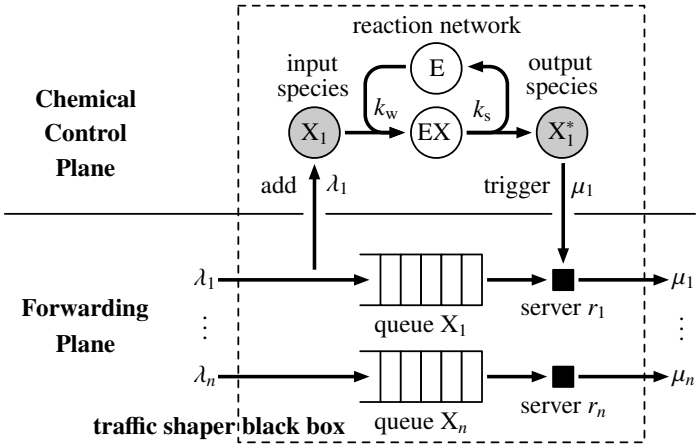


Figure 8: Chemical flow management architecture – The packet flow λ_1 is buffered in a classical FIFO queue while its server r_1 is triggered by a chemical reaction network. The enzymatic reaction implements a rate limitation policy.

Instead of processing *packet rates*, the chemical control plane may alternatively operate on *byte rates*. In this case, an incoming packet generates a quantity of molecules that corresponds to its length in bytes, whereas the server has to collect an amount of triggers equal to the length of the front packet in the queue before dequeuing it.

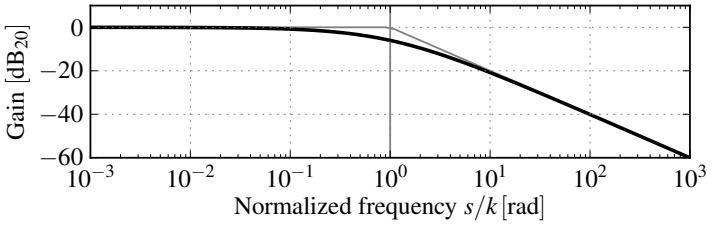
Although the abstract molecules in the chemical control plane represent packets (or bytes), they have no shape, meaning that they do not carry payload data. Since all molecule instances of the same species are identical, a typical implementation just has to store an integer value per species to keep track of its multiplicity. It is possible to either compile an abstract chemical network directly into a executable program or to use the reaction network to parameterize a generic reaction network simulator. Such a direct relation between the execution model and the abstract model allows a early and thorough analysis of the behavior already at design time.

Traffic smoothing. In the following, we present two examples how a simple chemical reaction network shapes packet flows. A simple unimolecular reaction between the input and output species as depicted in Figure 5, acts like a low-pass filter to a packet flow. This can be shown by converting the differential equation in (3) to the frequency domain. The transfer function below has low-pass characteristics with a cut-off frequency at the reaction constant k (see Figure 9(a)).

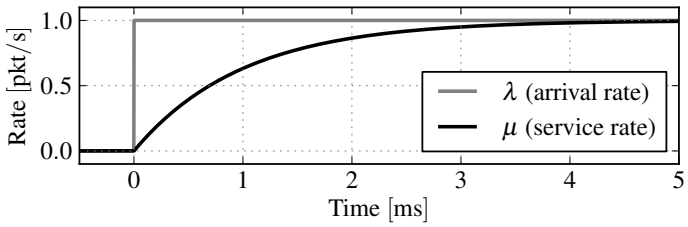
$$F(s) = \frac{\mu(s)}{\lambda(s)} = \frac{k}{s+k} \tag{10}$$

This means that bursts with high frequency components are smoothed. The step response of a LoMA-scheduled queue (depicted in Figure 9(b)) illustrates that the service rate exponentially approximates the arrival rate.

Such a low-pass filter applied at the ingress point of the network leads to less chaotic behavior of traffic patterns in the network. However, this does not come for free, as the filter imposes the packets with a delay of $d = 1/k$ and requires memory to buffer the packets: The mean fill level grows proportionally with the delay and the flow rate: $\hat{x} = \lambda/k$.



(a) Transfer function



(b) Step response

Figure 9: A LoMA-scheduled queue acts to a packet flow as a low-pass filter – Transfer function and step response of the reaction network in Figure 5.

Rate limiting. Unlike a traditional M/M/1 queue with a fixed service rate, a LoMA-scheduled queue exhibits an unbounded service rate, which follows the arrival rate. Our next showcase is a reaction network that is able to limit the rate of the packet flow.

Rate-limiting reactions are very common in biological systems in the form of enzymatic reactions, such as the one in the chemical control plane of Figure 8. Intuitively this rate limiter works as follows: Substrate molecules X are generated with the arrival rate λ. Enzyme molecules can be thought of as tokens: they are either in free form (E) or bound as enzyme-substrate complex (EX). The more enzymes are bound, (i.e., the less free enzymes there are), the slower will the transmission rate μ grow for an increasing arrival rate λ. We can also consider E as counting semaphore for which a data packet X has to "wait" before resting in EX for a short time. The semaphore is "signaled" when the packet is eventually sent.

According to Kirchhoff's current rule, the influx and efflux of species EX are equal at equilibrium: $k_w x_X x_E = k_s x_{EX}$. Since the total number of enzyme tokens is constant ($x_E + x_{EX} = e_0 = \text{const.}$) we obtain the *Michaelis-Menten equation* (Michaelis & Menten, 1913; van Slyke & Cullen, 1914; Teich, 1992), which expresses the transmission rate μ with respect to the fill level x_X of queue X:

$$\mu = v_{max} \frac{x_X}{K_M + x_X} \tag{11}$$

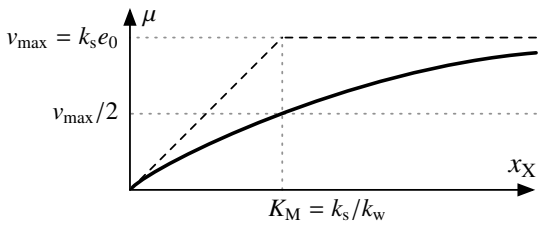


Figure 10: Saturation curve of the enzymatic reaction network.

Figure 10 depicts the resulting hyperbolic saturation curve. The constant $K_M = k_s/k_w$ specifies the fill level of X at which half of the maximal rate $v_{max} = k_s e_0$ is reached. For an empty queue ($x_X \to 0$) the rate limiter behaves like a unimolecular reaction with rate v_{max}/K_M whereas for a high fill level ($x_X \to \infty$) the transmission rate converges to the limit v_{max}. Obviously, the system is only stable if the offered *load* $\rho = \lambda/v_{max}$ is lower than 1. In other words, the steady-state fill level grows hyperbolically with the load and reaches infinity for $\rho \to 1$:

$$\hat{x}_X = K_M \frac{\rho}{1-\rho} \tag{12}$$

The mean latency of a packet through a chemical reaction network is generally calculated as the sum of the inverse egress reaction rates along a packet's path. In our enzymatic rate limiter, the average steady-state latency $T = T_X + T_{EX}$ of a packet through species X and EX is

$$T = \overbrace{K_M \frac{1}{v_{max} - \lambda}}^{T_X} + \overbrace{\frac{1}{k_s}}^{T_{EX}} \tag{13}$$

In the limit $k_2 \to \infty$ and $K_M \to 1$, the enzymatic rate limiter approaches the behavior of an M/M/1 queue ($T = 1/(v_{max} - \lambda)$).

4.3 Dynamic Queueing-Network Topologies

So far we demonstrated static queueing networks where abstract queues and servers were "installed" and "wired" permanently in each node. In this section, we extend this model aiming at dynamically changing the topology of the queueing network, meaning that queue instances and their flow relation can be generated on the fly. We therefore extended the Fraglets language (Tschudin, 2003) – an executable string and multiset rewriting system for efficient packet processing – by our LoMA scheduler, such that this simple platform is able to run "chemical networking protocols".

Fraglets. In Fraglets, each packet (or molecule) is a string of symbols over a finite alphabet Σ. Such a string is called a *fraglet*, because it represents a small fragment of data or in-network computation. The first symbol of the fraglet defines the string rewriting operation applied to it by the virtual chemical machine. Thus the header symbol can be thought of an assembler instruction. For example, the fraglet `[fork a b c d]` transforms itself and splits into the two fraglets `[a c d]` and `[b c d]`. The following list shows some of these instructions and their actions:

$$\begin{aligned}
\texttt{[match}\ \sigma\ \Phi\texttt{]} + \texttt{[}\sigma\ \Omega\texttt{]} &\longrightarrow \texttt{[}\Phi\ \Omega\texttt{]} \\
\texttt{[fork}\ \alpha\ \beta\ \Omega\texttt{]} &\longrightarrow \texttt{[}\alpha\ \Omega\texttt{]} + \texttt{[}\beta\ \Omega\texttt{]} \\
\texttt{[nop}\ \Omega\texttt{]} &\longrightarrow \texttt{[}\Omega\texttt{]} \\
{}_i\texttt{[send}\ k\ \Omega\texttt{]} &\longrightarrow {}_k\texttt{[}\Omega\texttt{]} \quad \text{if } (i,k) \in E
\end{aligned}$$

The wildcards $\alpha, \beta, \sigma \in \Sigma$ denote arbitrary symbols, whereas $\Phi, \Omega \in \Sigma^*$ represent symbol strings and $i, k \in \mathcal{V}$ are network nodes. Strings that start with `match` or any passive identifier, such as `X`, `Y`, or `Z`, are in their *normal form*. The `match`-instruction can be used to join two fraglets by concatenating the second to the first after removing the processed headers. Subsequent instructions immediately reduce the product

further until they again reach their normal form. For example, the two fraglets `[match X send 2 X]` and `[X data]` in node 1 imply the following reaction:

$$
\begin{aligned}
{}_1[\texttt{match X send 2 X}] + {}_1[\texttt{X data}] & \\
\longrightarrow {}_1[\texttt{send 2 X data}] & \\
\text{(transmission to neighbor 2)} \quad \longrightarrow {}_2[\texttt{X data}] &
\end{aligned}
$$

Rate-Limiter implementation in Fraglets. Such a chemical language allows us to "program" the reaction graph. Molecules now have a structure, meaning that they can *contain* information such as piggybacked user data. The matching tag (X, Y, Z, ...) identifies the queue into which a fraglet is inserted. There is one queue for all fraglets starting with `[match X...]` and a complementary queue into which fraglets starting with `[X ...]` are inserted. These queues are scheduled according to the law of mass action and thus, the behavior of the dynamically created packet flow is chemically controlled and can be analyzed by the tools presented before.

As an example, we provide the Fraglets code that evokes the enzymatic rate limiter in Figure 8. We install e_0 fraglets representing the enzyme E:

$$\texttt{[weight } k_\text{w} \texttt{ match X EX]} \tag{14}$$

Such a free enzyme reacts with a data packet `[X ...]` containing arbitrary payload and yields the bound enzyme EX as depicted in Figure 11. The following persistent fraglet represents reaction r_s:

$$\texttt{[weight } k_\text{s} \texttt{ matchp EX weight } k_\text{w} \texttt{ match X EX]} \tag{15}$$

It reacts with the bound enzyme, splits off the product with the payload `[X* ...]`, and regenerates the free enzyme E.

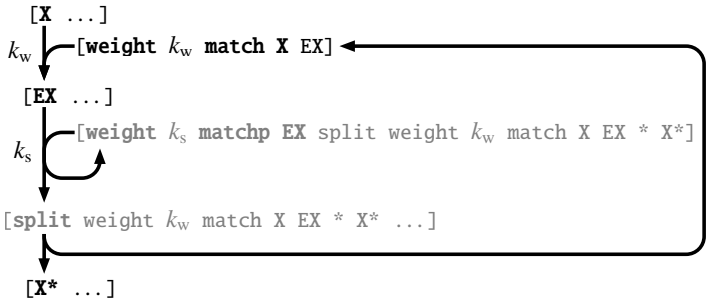


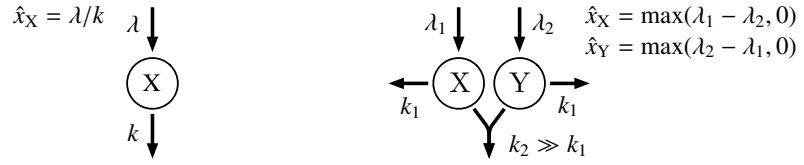
Figure 11: Fraglets implementation of the rate limiter – See Figure 8 for the model and a statically wired "chemical control plane" implementation.

Discussion. Unlike in our first execution model, where the flow interactions are determined at design time, a fraglet decides itself into which queue it is sorted. After a packet is serviced, its header is treated as code, which allows a packet to determine its route through the network akin to active networking.

The tight relation between abstract chemical model and execution layer however remains, meaning that in bottom-up direction, a mathematical model of the behavior of a Fraglets program can be generated automatically, and in top-down direction, that a queueing network designed in the abstract model can be realized in Fraglets.

4.4 Motifs — Chemical Design Patterns

Like all design processes, the design of a queueing network cannot be automated. But we developed a number of generic design patterns, which in the chemical world are simple reaction network *motifs* that can easily be combined. We already discussed the rate-limiting motif in Section 4.2. In the following, we will introduce two additional motifs.



(a) Converts rate λ to fill level $\hat{x}_X$

(b) Computes the difference between two rates λ_1 and λ_2

Figure 12: Reaction network motifs.

The rate-to-fill-level conversion motif depicted in Figure 12(a) makes sure that the steady-state fill level of queue X is proportional to the influx rate: $\hat{x}_X = \lambda/k$. The reaction constant k serves as conversion factor.

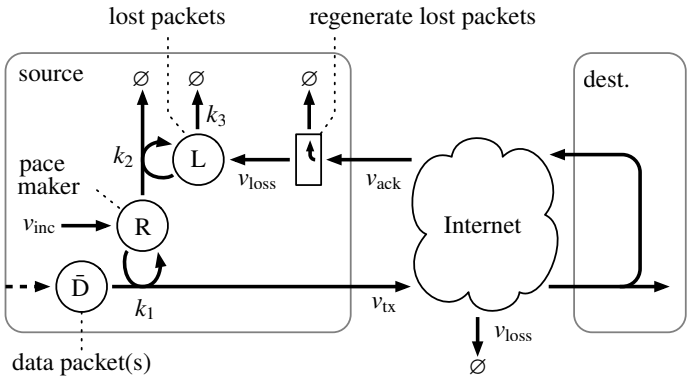
Figure 12(b) shows a motif that combines two instances of the previous one in order to compute the difference between the arrival rates λ_1 and λ_2 of the two queues X and Y. The bimolecular efflux reaction atomically removes a packet from both queues with a high rate. Consequently, in steady state, only one of the queues contains packets, and its fill level represents the difference of the two arrival rates.

We developed other motifs for various purposes, ranging from arithmetic computation of fill-levels, to communication patterns such as anycast, neighborhood discovery, etc.

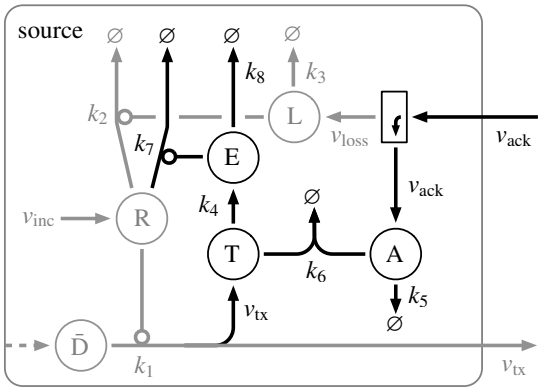
5 C3A — A Chemical Congestion Control Algorithm

So far, we studied single LoMA-scheduled queues and simple flow-filtering patterns. In this section, we demonstrate the straightforward combination of queues to "schedule" segments of a transport protocol with the purpose of controlling the caused congestion. Congestion control is a novel application of queuing theory, made possible by the law-of-mass-action discipline. To this end, we re-implement the additive increase / multiplicative decrease mechanism of TCP Reno's congestion avoidance mode, originally proposed by Jacobson (1988). It is not our aim to cover all features of TCP such as fast start, but to demonstrate how a flow of data packets is throttled by a simple queueing network such that the emitted data stream is fair to TCP streams.

Figure 13(a) shows the reaction network of our chemical congestion control algorithm C3A. Arriving data packets are placed into queue D. The transmission rate is controlled by the quantity of pace maker molecules R: $v_{tx} = k_1 x_R x_D$, according to the law of mass action. The number of pace maker molecules is continuously increased at rate v_{inc}, which mimics the additive (here: linear) increase mechanism. Before being transmitted the packets are tagged with a continuously increasing sequence number. Based on a gap in the sequence number of received acknowledgments the source node is able to regenerate the lost packets and insert them into queue L. Such a lost packet catalyzes the destruction of pace maker molecules through reaction r_2, which leads to an exponential decay of R-molecules and a corresponding decrease of the transmission rate. The duration of the decay however is limited as lost molecules are also



(a) C3A, reacting to packet loss



(b) C3A$^+$, additionally reacting to RTT variations

Figure 13: Reaction network of two chemical congestion control algorithms – The reaction network in (a) is reused in (b) (shaded in grey) and extended by new species and reactions (black) in order to sense and react to round-trip-time (RTT) variations.

decayed by reaction r_3.

5.1 The Reaction Graph as Design Instrument

The reaction graph (as the one depicted in Figure 13(a)) is a very useful tool in the design phase of flow management algorithms. It conveys information in a much more intuitive way that an implementation of the same functionality in an imperative procedural language, such as C. For instance, the linear increase mechanism in our example is clearly visible as the inflow of R-molecules. We can also easily recognize the feedback nature of congestion control when looking at the part of the network where the lost packets in L invoke the death of pace makers R. Similar to electronic circuits where currents control the flow of other currents in transistors, here packet flows control the flow of other packets via bimolecular reactions.

5.2 TCP-Fairness of C3A

Analytical treatment. According to Floyd and Fall (1999), Floyd, Handley, Padhye, and Widmer (2008), the transmission rate of TCP-fair packet streams should be proportional to $1/\sqrt{p_{\text{loss}}}$ where p_{loss} denotes the round trip packet loss probability between source and destination. With Kirchhoff's current law (see Section 3.3) we can easily proof that our queueing network satisfies this property: At equilibrium,

the fill level of the R- and L-queues do not change and their in- and out-flows are equal:

$$\overbrace{v_{\text{inc}}}^{\text{inflow of R}} = \overbrace{k_2 \hat{x}_{\text{R}} \hat{x}_{\text{L}}}^{\text{outflow of R}} \tag{16a}$$

$$\underbrace{\hat{v}_{\text{loss}}}_{\text{inflow of L}} = \underbrace{k_3 \hat{x}_{\text{L}}}_{\text{outflow of L}} \tag{16b}$$

According to the law of mass action, the transmission rate is $\hat{v}_{\text{tx}} = k_1 \hat{x}_{\text{R}} \hat{x}_{\text{D}}$; into this equation, we plug in $\hat{x}_{\text{R}}$ from (16a).

$$\hat{v}_{\text{tx}} = \frac{k_1 v_{\text{inc}} \hat{x}_{\text{D}}}{k_2 \hat{x}_{\text{L}}} \tag{17}$$

Next, we take $\hat{x}_{\text{L}}$ from (16b) and express the packet loss rate v_{loss} with respect to the loss probability: $\hat{v}_{\text{loss}} = \hat{v}_{\text{tx}} p_{\text{loss}}$.

$$\hat{v}_{\text{tx}} = \frac{k_1 k_3 v_{\text{inc}} \hat{x}_{\text{D}}}{k_2 \hat{v}_{\text{tx}} p_{\text{loss}}} \tag{18}$$

By multiplying both sides by $\hat{v}_{\text{tx}}$ and taking the square root we obtain a steady-state transmission rate of

$$\hat{v}_{\text{tx}} = \sqrt{\frac{k_1 k_3 r_{\text{inc}} \hat{x}_{\text{D}}}{k_2}} \cdot \frac{1}{\sqrt{p_{\text{loss}}}} \tag{19}$$

We determined optimal reaction constants by optimizing the behavior of the reaction network with respect to a short response time and a small overshoot. Were therefore linearized the ODEs of the system and determined the frequency response of the resulting LTI system. The obtained constants are $k_1 = 1/(\text{pkt}\,\text{s})$, $k_2 = 10^4/(\text{pkt}\,\text{s})$, $k_3 = 10^5/\text{s}$, and, as rate to increment the pace maker, $v_{\text{inc}} = 10^3\,\text{pkt/s}$.

OMNeT++ simulations. We empirically verified this theoretical dependence between packet loss and resulting transmission rate by OMNeT++ simulations. Figure 14 shows that the simulation results match the predicted transmission rate well for different packet loss probabilities.

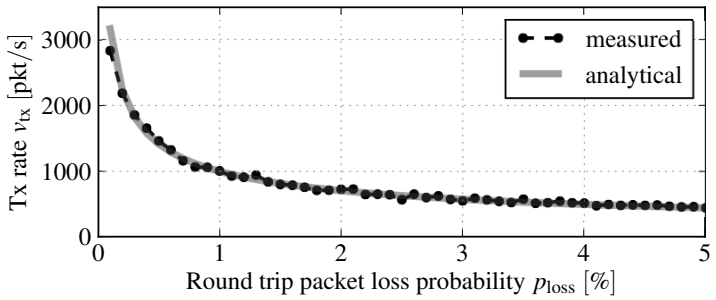


Figure 14: TCP-Fairness of C3A – The transmission rate drops with a square root relation with respect to the packet loss probability. We determined the values analytically from (19) and complemented them with empirical OMNeT++ simulation results, averaged over a transmission time of 10 s.

Figure 16(a) depicts the simulation results for two data streams that are sent over the same bandwidth-limited link; the bandwidth is $b = 1\,\text{MB/s}$ and the delay $d = 5\,\text{ms}$. The two plots on the top display the saw-tooth pattern of the transmission rate of both source nodes, which is caused by the linear increase / exponential decrease behavior of C3A.

5.3 C3A$^+$ — An Extended Version Additionally Reacting to RTT Variations

As the plot in the bottom of Figure 16(a) indicates, our chemical congestion control algorithm (C3A) is not able to fully exploit the link bandwidth. In order to improve its efficiency, we make use of a delay-based dual optimization strategy (Kelly, 2003; Kelly, Maulloo, & Tan, 1998). Instead of solely using lost packets as congestion indication, we *additionally* exploit the round-trip-time (RTT) variation. The RTT is increased by queues along the path that are filled during a congestion situation as shown in the bottom of Figure 15. As a consequence, the transmission rate v_{tx} rises while the rate of acknowledgments v_{ack} stagnates (see top of the figure).

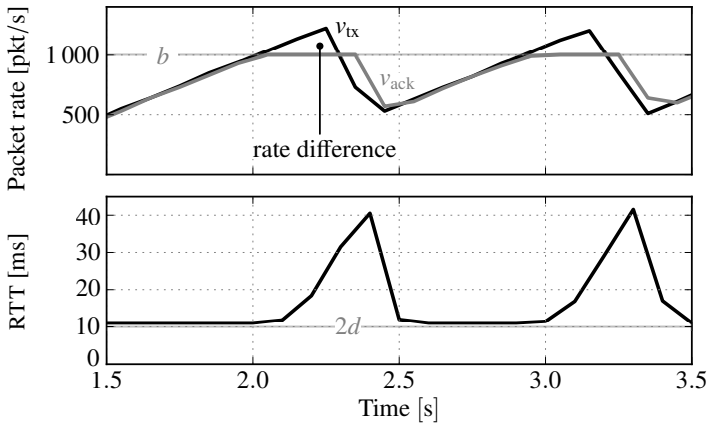
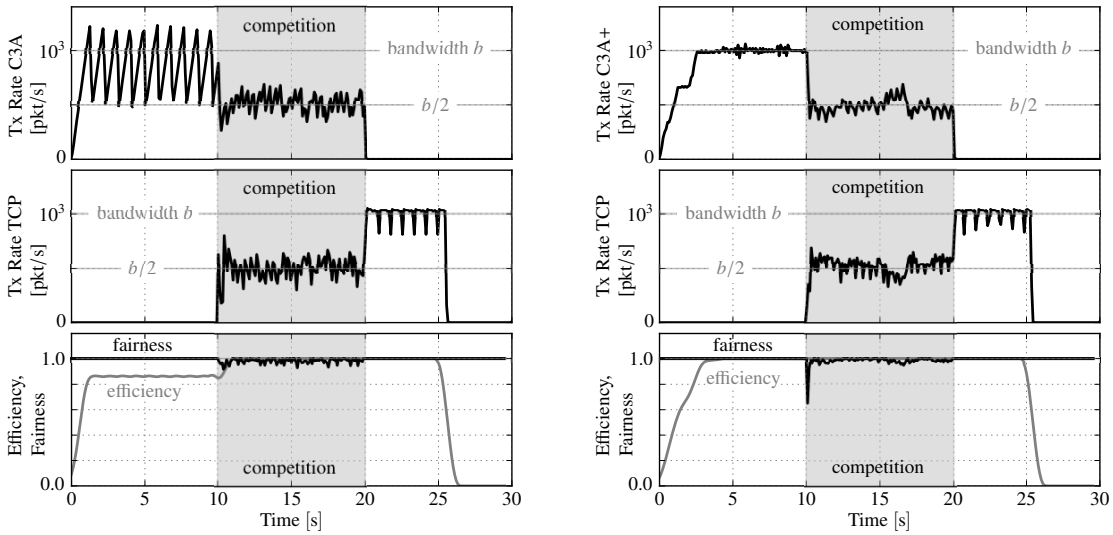


Figure 15: C3A-controlled packet stream – Observable difference between v_{tx} and v_{ack} if queuing delays accumulate due to congestion. This margin can be "chemically" detected and further processed.

Unlike TCP Vegas (Brakmo & Peterson, 1995), which measures the minimum round trip time (RTT) and lowers its transmission rate if the RTT increases, we do not compute traffic statistics symbolically with chemical reactions. Figure 13(b) depicts the reaction network of C3A$^+$, which measures the RTT variation indirectly by computing the difference between transmission and acknowledgment rate. For this purpose we make use of the rate difference motif discussed in Section 4.4. This motif appears in the bottom right corner of the reaction network in Figure 13(b). Without reaction r_6, the fill levels of the queues T and A would reflect the transmission rate and the acknowledgment rate, respectively. However, reaction r_6 builds the difference such that at equilibrium T is proportional to the excess transmission rate. We use this "signal" to decay the pace maker molecules R via E, as we did for packet losses.

OMNeT++ simulations. Figure 16(b) shows that the efficiency of the algorithm could be increased and that the response is much smoother than in our original design, although the response time is a bit slower (compare to Figure 16(a)). Note that because the algorithm is not aware of the minimal RTT, it slowly tries to exploit the capacity of the queues along the path such that eventually a packet will be lost. However, by being sensitive to RTT variations we could reduce the packet loss by 80%.

Summary. The aim of our "chemical" approach to congestion control is not to compete with existing advanced TCP implementations. We rather want to demonstrate the advantage of the mass-action-scheduling regime for the design and analysis of packet flow management algorithms. We showed that both primal and dual optimization strategies can be combined to an algorithm that is still TCP-fair.



(a) C3A, reacting to packet loss

(b) C3A$^+$, additionally reacting to RTT variations

Figure 16: Chemical congestion controlled packet stream in competition to a TCP-stream – Simulation in OMNeT++ with its native TCP Reno implementation: MSS = 10^3 B, delayed ACKs are disabled, Nagle's algorithm is enabled (Nagle, 1984), selective ACKs are disabled (Mathis, Mahdavi, Floyd, & Romanow, 1996), the advertised window is at the maximum of 64 kB. We measured the efficiency as the total transmission rate and the fairness by using Jain's fairness index (Jain, Chiu, & Hawe, 1984).

6 Discussion

Our account above of LoMA-based packet flow management has to be completed in several ways, with what we know today and what still has to be explored in future work.

Niche or Realm. In this paper, we showed two spots in a networking stack where LoMA scheduling can be applied. First, the enzymatic rate limiter is a classical traffic policing mechanism that we labeled with (c) in Figure 17. Second, the C3A algorithm could be turned into a full transport protocol that is TCP friendly by design (label (b) in the figure). The vanilla place to experiment with LoMA scheduling in an Internet is of course the application space (a).

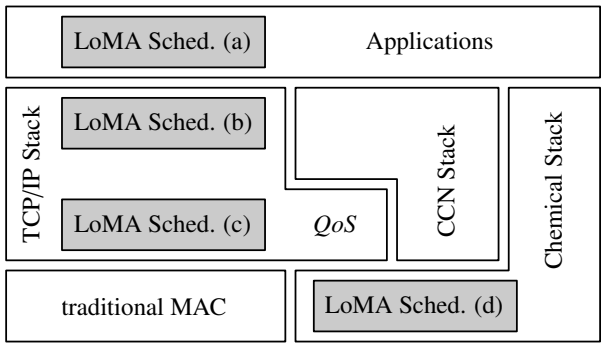


Figure 17: Architecture – Various places where to integrate LoMA scheduling into networking stacks.

All these spots might be worth exploring under the assumptions that widespread work-conserving scheduling of traditional MACs does work nicely enough, especially in an over-provisioned environment. However, to directly benefit from LoMA dynamics, it seems most interesting to investigate "chemical" links on which QoS stacks can be built. We also mention content centric networking (Jacobson, 2006; Jacobson et al., 2009) here because we feel that this flow-aware approach, which bridges application level objects with forwarding level concerns, would naturally benefit from dynamics control built into its framework (d).

A possible use case of LoMA scheduling is delay-neutral, scalable forwarding in QoS networks. We pointed out in Table 2 that LoMA forwarding incurs a constant delay. One obvious application is in the field of QoS, where the chemical setting guarantees that the end-to-end delay is constant and the jitter is kept small, independent of the activity of other streams. Assume a sufficiently dimensioned node that keeps per-flow forwarding state, each flow being equipped with its own LoMA queue with low-pass behavior in order to keep the jitter small. If the individual flows are now multiplexed through another LoMA queue into trunks (in order to reduce forwarding state in the core), the low end-to-end jitter remains. That is, the forwarding node multiplexes parallel streams without any crosstalk by increasing the delay by a constant amount, determined solely by the reaction constant of the multiplexing reaction.

Demarcation. Formulated as a hypothesis, we think that *best-effort traffic* is intrinsically incompatible with LoMA scheduling (a statement that we would have to capture and examine in formal ways). Strict packet flow prioritization, for example, is hardly achievable because the pressure generated by the growing fill level of a low-priority or best-effort queue would inevitably lead to its prioritization. This is both a limitation and a useful architectural insight: trying to embed best-effort in our framework is probably not worthwhile, and running LoMA flows over the current Internet will only exploit part of its potential. However, *parallel worlds* can be conceived where resource boundaries outside LoMA scheduling are engineered that assign left-over buffer and bandwidth resources to the best-effort packet class, as well as well-defined transit points to enter and leave the space of LoMA-managed packets. Finally, one should not credit LoMA scheduling to lead to "better" flow behavior *per se*: *Fairness*, for example, still remains a characteristic that has to be programmed (which is feasible in the LoMA context, as we showed with our C3A$^+$ algorithm). The low-pass filter effect, however, might be beneficial insofar as it mitigates the on-set impact of traffic bursts of end nodes to core nodes and servers under DoS attacks.

Implementation Issues and Future Work. The queueing model presented in this paper makes four assumptions that simplify its treatment and that we have to discuss here: First, we based our flow-level analysis on packet rates. But since the link capacity is physically given in terms of bytes per second, we usually want to consider the byte rate of packet flows instead. For the LoMA scheduler this is more tricky in variable size packet networks because the service time may change on-the-fly while a link is busy sending the previous packet. We sketched a possible solution to take the packet size into account in Section 4.2.

Second, real links impose a packet delay additional to the waiting time of a queue. In the fluid model, this would lead to a Delay Differential Equation (DDE) system, which is generally hard to analyze. If we are only interested in the steady-state behavior of a queueing network (e.g., when analyzing the equilibrium fill-level distribution), a link with delay d can be modeled as an additional LoMA-scheduled queue with a reaction constant of $k = 1/d$. Such a virtual queue simulates the delay of the link but keeps the mathematical model free of delay terms.

Third, the message complexity of "chemical" protocols will be higher than traditionally. The *Disperser* protocol, for example, uses the packet rate to convey information; more messages are exchanged than in the comparable *Push-Sum* protocol. However, this is only an issue if the packets cross a real link between two queues. For queueing networks such as the enzymatic rate limiter, packets are not sent remotely but only used to control other packet flows locally. On the other hand, TCP demonstrates that in order to perform rate control, frequent acknowledgment packets is the price to pay for having a reliable feedback signal from the network.

Fourth, we assumed that the queue capacity is infinite. In reality, all queues are bound and some drop behavior has to be specified. For the LoMA-scheduling discipline this means that the maximum service rate automatically has an upper bound, too. This makes sense, as a CPU must be able to execute the scheduled events in time. Thus in LoMA scheduling, memory limits are directly linked to the performance limits of the executing machinery. This allows for controlling one with the other, and makes it possible to dimension the system in advance.

7 Conclusions

Computer networking is the art of combining logical functionality with dynamic behavior in a distributed setting: On one hand, it has to logically organize a distributed computation such that useful side effects like coherent routing state or accurate content copies emerge from the exchange of data packets despite the imponderability of communication links and failing nodes. On the other hand, it has to orchestrate delay by deferring packets; otherwise network dynamics will grow out of control and will make all logical attempts to provide functionality futile. Law-of-Mass-Action scheduling is a low-level fixture on packet delay that introduces useful restrictions over the current work-conserving scheduling scheme: It separates forwarding from the (work-conserving) timing and makes time control subject to a programmable environment. When this programming is done with an artificial chemistry, rich and yet analyzable dynamics can be engineered on the flow-level without the need to micro-manage the delays individually. We hope that our work contributes to turning delay management from a "black art" into an engineering discipline.

Acknowledgments

Markus Fidler came up with the useful observation that todays wireless MAC protocols are non-work conserving, too. We gratefully acknowledge Per Gunningberg's helpful editing comments. All inaccuracies and errors remain our fault. This work has been supported by the Swiss National Science Foundation through SNF Project Self-Healing Protocols (2000201-109563).

References

Abrash, H. I. (1986). Studies concerning affinity. *Journal of Chemical Education*, *63*, 1044–1047. English translation of Waage & Guldberg, 1864. doi:10.1021/ed063p1044

Anderson, D., Craciun, G., & Kurtz, T. (2010). Product-form stationary distributions for deficiency zero chemical reaction networks. *Bulletin of Mathematical Biology*, *72*, 1947–1970. doi:10.1007/s11538-010-9517-4

Brakmo, L., & Peterson, L. (1995). TCP Vegas: End to end congestion avoidance on a global Internet. IEEE *Journal on Selected Areas in Communications*, *13* (8), 1465–1480. doi:10.1109/49.464716

Dittrich, P., & Speroni di Fenizio, P. (2007). Chemical organization theory. *Bulletin of Mathematical Biology*, *69* (4), 1199–1231. doi:10.1007/s11538-006-9130-8

Dittrich, P., Ziegler, J., & Banzhaf, W. (2001). Artificial chemistries - a review. *Artificial Life*, *7* (3), 225–275. doi:10.1162/106454601753238636

Elf, J., & Ehrenberg, M. (2003). Fast evaluation of fluctuations in biochemical networks with the linear noise approximation. *Genome Research*, *13* (11), 2475–2484. doi:10.1101/gr.1196503

Eryilmaz, A., Srikant, R., & Perkins, J. R. (2001). Throughput-optimal scheduling for broadcast channels. In *Proc.* SPIE (Vol. 4531, pp. 70–78).

Feinberg, M. R. (1972). Complex balancing in general kinetic systems. *Archive for Rational Mechanics and Analysis*, *49* (3). doi:10.1007/BF00255665

Fell, D. A. (1997). *Understanding the control of metabolism*. London: Portland Press.

Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2008). TCP friendly rate control (TFRC): protocol specification. RFC 5348 (Proposed Standard). Internet Engineering Task Force. IETF. Retrieved from Internet Engineering Task Force: http://www.ietf.org/rfc/rfc5348.txt

Floyd, S., & Fall, K. (1999). Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM *Transactions on Networking*, *7* (4), 458–472. doi:10.1109/90.793002

Gadgil, C., Lee, C. H., & Othmer, H. G. (2005). A stochastic analysis of first-order reaction networks. *Bulletin of Mathematical Biology*, *67* (5), 901–946. doi:doi:10.1016/j.bulm.2004.09.009

Gelenbe, E. (2008). Network of interacting synthetic molecules in steady state. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, *464* (2096), 2219–2228. doi:10.1098/rspa.2008.0001

Gibson, M. A., & Bruck, J. (2000). Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry A*, *104* (9), 1876–1889. doi:10.1021/jp993732q

Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, *81* (25), 2340–2361. doi:10.1021/j100540a008

Gillespie, D. T. (1992). A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, *188* (1–3), 404–425. doi:10.1016/0378-4371(92)90283-V

Gillespie, D. T. (2000). The chemical Langevin equation. *Journal of Chemical Physics*, *113* (1). doi:10.1063/1.481811

Gillespie, D. T. (2002). The chemical Langevin and Fokker-Planck equations for the reversible isomerization reaction. *Journal of Physical Chemistry A*, *106* (20), 5063–5071. doi:10.1021/jp0128832

Gómez-Uribe, C. A., & Verghese, G. C. (2007). Mass fluctuation kinetics: capturing stochastic effects in systems of chemical reactions through coupled mean-variance computations. *Journal of Chemical Physics*, *126* (2). doi:10.1063/1.2408422

Heinrich, R., & Schuster, S. (1996). *The regulation of cellular systems*. Springer.

Hofmeyr, J. H. S. (2001). Metabolic control analysis in a nutshell. In T. M. Yi, M. Hucka, M. Morohashi & H. Kitano (Eds.), *Proc. 2nd international conference on systems biology* (pp. 291–300). Madison, WI, USA: Omnipress.

Horn, F. (1973a). On a connexion between stability and graphs in chemical kinetics. I. Stability and the reaction diagram. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, *334* (1598), 299–312. doi:10.1098/rspa.1973.0093

Horn, F. (1973b). On a connexion between stability and graphs in chemical kinetics. II. Stability and the complex graph. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, *334* (1598), 313–330. doi:10.1098/rspa.1973.0094

Horn, F., & Jackson, R. (1972). General mass action kinetics. *Archive for Rational Mechanics and Analysis*, *47* (2). doi:10.1007/BF00251225

Ingalls, B. (2004). A frequency domain approach to sensitivity analysis of biochemical networks. *Journal of Physical Chemistry B*, *108* (3), 1143–1152.

Jacobson, V. (1988). Congestion avoidance and control. In *Proc. symposium on communications architectures and protocols* (pp. 314–329). Stanford, CA, USA: ACM. doi:10.1145/52324.52356

Jacobson, V. (2006). A new way to look at networking. Retrieved 8 October 2010, from http://video.google.com/videoplay?docid=-6972678839686672840#

Jacobson, V., Smetters, D. K., Thornton, J. D., Plass, M. F., Briggs, N., & Braynard, R. (2009). Networking named content. In *Proc. 5th ACM international conference on emerging networking experiments and technologies (CONEXT 2009)* (pp. 1–12).

Jahnke, T., & Huisinga, W. (2007). Solving the chemical master equation for monomolecular reaction systems analytically. *Journal of Mathematical Biology*, *54*, 1–26.

Jain, R. K., Chiu, D.-M. W., & Hawe, W. R. (1984). *A quantitative measure of fairness and discrimination for resource allocation in shared computer systems* (Research Report No. TR-301).

Kamimura, K., Hoshino, H., & Shishikui, Y. (2008). Constant delay queuing for jitter-sensitive IPTV distribution on home network. In *Ieee global telecommunications conference (ieee globecom 2008)*.

Kelly, F. P. (2003). Fairness and stability of end-to-end congestion control. *European Journal of Control*, *9*, 159–176.

Kelly, F. P., Maulloo, A. K., & Tan, D. K. H. (1998). Rate control for communication networks: Shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, *49* (3). JSTOR: 3010473. Retrieved from http://www.jstor.org/stable/3010473

Kempe, D., Dobra, A., & Gehrke, J. (2003). Gossip-based computation of aggregate information. In *Proc. 44th annual IEEE symposium on foundations of computer science* (pp. 482–491).

Le Boudec, J.-Y., & Thiran, P. (2004). *Network calculus*. Lecture Notes in Computer Science. Springer.

Le Pocher, H., Leung, V., & Gillies, D. (1999). An application- and management-based approach to ATM scheduling. *Telecommunication Systems*, *12*, 103–122.

Low, S. H. (2003). A duality model of TCP and queue management algorithms. IEEE/ACM *Transactions on Networking*, *11* (4), 525–536.

Mairesse, J., & Nguyen, H.-T. (2009). Deficiency zero Petri nets and product form. *arXiv.org cs.DM*, *0905.3158v2*.

Mathis, M., Mahdavi, J., Floyd, S., & Romanow, A. (1996). TCP selective acknowledgment options. RFC 2018 (Proposed Standard). Internet Engineering Task Force. IETF. Retrieved from Internet Engineering Task Force: http://www.ietf.org/rfc/rfc2018.txt

McQuarrie, D. A. (1967). Stochastic approach to chemical kinetics. *Journal of Applied Probability*, *4* (3), 413–478.

Meyer, T., & Tschudin, C. (2009). Chemical networking protocols. In *Proc. hot topics in computer networks (hotnets-VIII)*. ACM.

Michaelis, L., & Menten, M. (1913). Die Kinetik der Invertinwirkung. *Biochemische Zeitschrift*, *49*, 333–369. Retrieved 13 August 2010, from `http://web.lemoyne.edu/~giunta/menten.html`

Misra, V., Gong, W.-B., & Towsley, D. (2000). Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED. SIGCOMM *Computer Communication Review*, *30*, 151–160.

Nagle, J. (1984). Congestion control in IP/TCP internetworks. RFC 896. Internet Engineering Task Force. IETF. Retrieved from Internet Engineering Task Force: `http://www.ietf.org/rfc/rfc896.txt`

Perelson, A. S., & Oster, G. F. (1974). Chemical reaction dynamics part II: Reaction networks. *Archive for Rational Mechanics and Analysis*, *57* (1), 31–98. doi:`10.1007/BF00287096`

Reder, C. (1988). Metabolic control theory: a structural approach. *Journal of Theoretical Biology*, *135* (2), 175–201. doi:`10.1016/S0022-5193(88)80073-0`

Seong, K., Narasimhan, R., & Cioffi, J. (2006). Queue proportional scheduling in gaussian broadcast channels. In *Ieee international conference on communications* (Vol. 4, pp. 1647–1652).

Shakkottai, S., & Srikant, R. (2002). How good are deterministic fluid models of Internet congestion control? In *Proc. 21st annual joint conference of the* IEEE *computer and communications societies (*INFOCOM *2002)* (Vol. 2, pp. 497–505).

Tassiulas, L., & Ephremides, A. (1992). Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, *37* (12), 1936–1949.

Teich, M. (1992). *Documentary history of biochemistry, 1770–1940*. Rutherford, NJ, USA: Fairleigh Dickinson University Press.

Tomioka, R., Kimura, H., Kobayashi, T. J., & Aihara, K. (2004). Multivariate analysis of noise in genetic regulatory networks. *Journal of Theoretical Biology*, *229* (4), 501–521. doi:`10.1016/j.jtbi.2004.04.034`

Tschudin, C. (2003). Fraglets - a metabolistic execution model for communication protocols. In *Proc. 2nd annual symposium on autonomous intelligent networks and systems (*AINS*)*.

Ullah, M., & Wolkenhauer, O. (2009a). Investigating the two-moment characterisation of subcellular biochemical networks. *arXiv.org q-bio.SC*, *0809.0773v3*.

Ullah, M., & Wolkenhauer, O. (2009b). Stochastic approaches in systems biology. *Wiley Interdisicplinary Reviews: Systems Biology and Medicine*.

van Kampen, N. G. (1976). The expansion of the master equation. *Advances in Chemical Physics*, *34*.

van Kampen, N. G. (2007). *Stochastic processes in physics and chemistry* (3rd ed.). Elsevier.

van Mieghem, P. (2006). *Performance analysis of communications networks and systems*. Cambridge University Press.

van Slyke, D. D., & Cullen, G. E. (1914). The mode of action of urease and of enzymes in general. *Journal of Biological Chemistry*, *19*, 141–180.

Waage, P., & Guldberg, C. M. (1864). Studies concerning affinity. *Forhandlinger: Videnskabs - Selskabet i Christiania*, *35*.